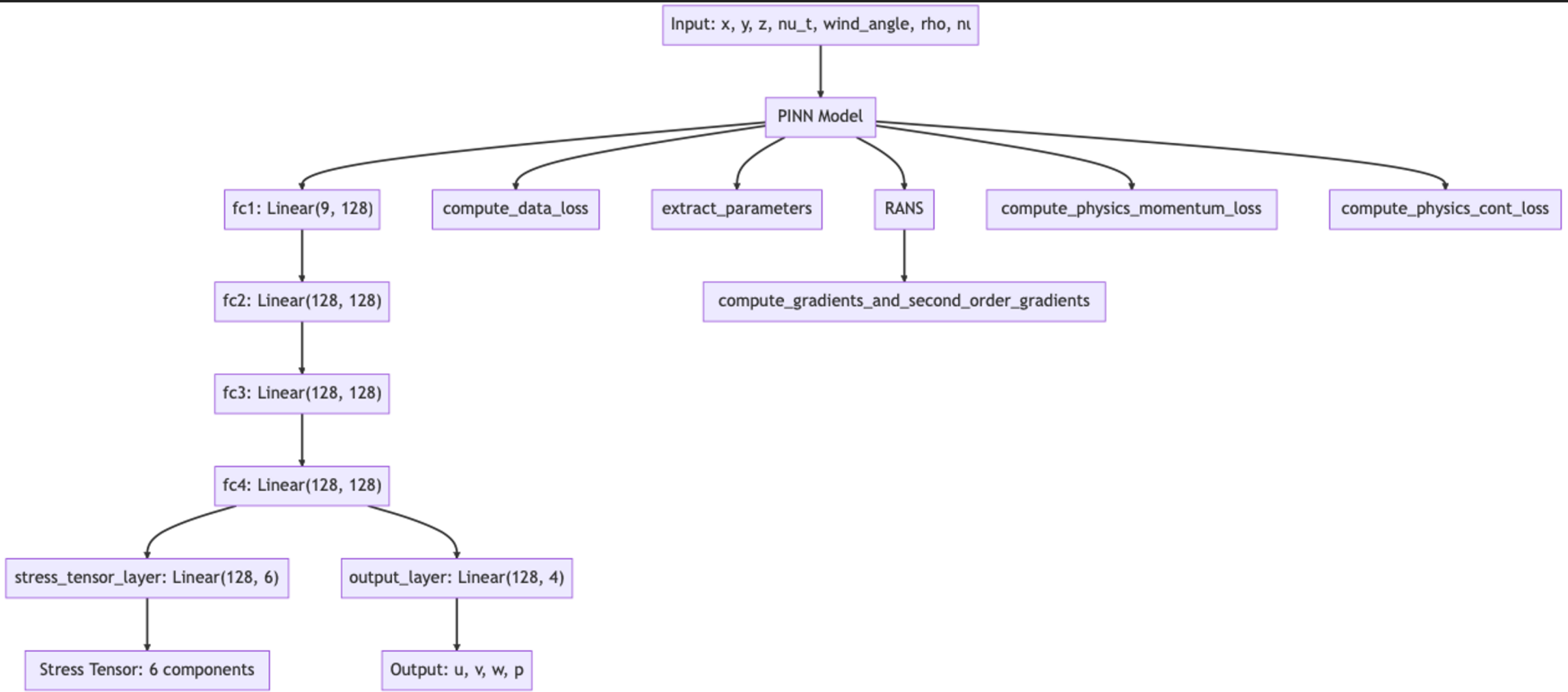
PM002 – 18 October 2022
Update on PINNs

Application to Urban Wind Field Dispersion Studies

Armand Albergel, Claude Guet & Ameir Shaa

Neural Network Architecture



Some Comment to Address

- **Input**: Limiting the input variables to x, y, z, $\cos(\theta)$ and $\sin(\theta)$ should be enough in this phase of the study – Ameir agrees, Claude ?
- Nu-t (ν_t) is associated with turbulence production/destruction and, therefore, should be considered an output (in the RANS model, it is proportional to the square of the kinetic turbulent energy and inversely proportional to turbulent dissipation rate y, ($\nu_t = C\frac{k^2}{\varepsilon}$). Rho and nu-molecular are constants in this context. While we need them to verify RANS, it doesn't seem necessary to complicate the input – Ameir & Claude agree
- **Output**: May we consider u, v, w, p, ν_t? The Reynolds stress tensor is not needed I think as we consider only the trace (k) and ν_t is sufficient for RAND PDE. ν_t should be directly be used in dispersion model – Ameir agrees somewhat, although it would be a nice to have a comparison with DNS data, Claude

- Reynolds stress tensor (no learning data and not very useful) – Ameir agrees (and it is very difficult to do so)

Some Parameters

Infinite epochs - instead the criteria for stopping is loss_{n} - loss_{n-1} < ε for 10 consecutive epochs where n is the epoch number and ε = 1E-5 (user defined)

We have the data for 7 angles, [0, 30, 60, 90, 120, 150, 180] in degrees

We concatenate the data for angles = [0, 30, 60, 120, 150, 180] and then take 80% of the dataset with random seed = 42 for training and 20% for testing

By using the whole dataset we hope to make the NN learn about wind angle such that the parameters become functions of the wind angle

We also would like the stress tensor to be learnt so that we do not have to assume models involving turbulence (ie. model free approach)

Then using the trained neural network we predict the data for angle = 90

Some Comments

While the criterion of monitoring loss reduction over 10 consecutive epochs is a reasonable approach, it's important to tailor the stopping criterion to the specific requirements of your task and dataset. My understanding is that some datasets may require more or fewer epochs to reach a suitable solution. Adapting the stopping criterion accordingly can help account for this variability – Ameir & Claude agree

I understand that for the learning phase:

- the whole Code_Saturne 3D output field is used to select the learning data set (80%) and the testing data set (20%) The selection of 20% is randomly done (with random seed = 42 ?) – yes this is correct
- I am afraid that the 20% may miss or under-represent “interesting” part of the flow (around obstacles / wake …) – it may be the case but this is why we have randomness as well as a seed number so we can reproduce any results we deem unfit or find weird

Some Comments

Please confirm:

- the normalization of the different physical data - yes this is correct, data is normalized according to a standard gaussian. This is opposed to a min-max normalization (eg [-1, 1] and etc); reasoning in the next slide.
- the shuffling done to ensure that the training data is presented in a random order during each epoch - yes this is correct
- which activation function selected (tanh? as Bowen?) – no, ReLu (defined as f(x) = max(0, x) is used instead; reasoning in next slide
- May be see the influence in selecting another angle? – a must!!

Why a Standard Gaussian?

1. **Centered Data**: Z-score normalization centers the data around zero. This can be beneficial for algorithms like gradient descent, as it ensures that all features contribute equally to the model's prediction, preventing any single feature from disproportionately influencing the model.
2. **Equal Variance**: By scaling the data to have a standard deviation of 1, it ensures that all features have the same scale of variance. This can be particularly important for algorithms that are sensitive to the scale of the input features, such as support vector machines or k-means clustering.
3. **Less Sensitive to Outliers**: Z-score normalization is less sensitive to outliers than min-max scaling. In min-max scaling, if there are outliers in the data, they can cause the majority of the scaled values to be squeezed into a small interval. Z-score normalization, on the other hand, is less affected by this issue.
4. **Better for Data with Unknown Boundaries**: If the maximum and minimum values of the features are not known in advance or can change, min-max scaling can be problematic. Z-score normalization does not have this limitation.
5. **Preserves Relative Distances**: Z-score normalization preserves the relative distances between data points, which can be important for algorithms that rely on distance metrics.

- However, it's worth noting that the choice of scaling method can be problem-specific. In some cases, min-max scaling might be more appropriate, especially if the data has a known and fixed range. Additionally, for neural networks with activation functions like sigmoid or tanh that squash their input into a small range, min-max scaling might be more suitable.

Why not tanh?

- We need to take double derivatives in our RANS loss function
- Consider the function f(x) = tanh(x) – computationally more expensive than ReLu.
- If the domain of f(x) is [-1,1], the range of f(x) ~ (-0.75, 0.75)
- First derivative of f(x) = sech^2(x) has a range of (0,1] for the domain [-1,1] – this is limiting and squashes your values. (ie. the points are now closer to one another).
- There are benefits of using tanh as the activation though.. It depends on the problem.

Some Comments

- Adam, short for "Adaptive Moment Estimation," is an extension of the stochastic gradient descent (SGD) optimization method and is known for its efficiency and effectiveness in a wide range of tasks. - indeed
- Good scores: risk of overfitting? – yes, I agree. LGBFS (Limited-memory BFGS) is a better optimizer for our problem. A very nice paper - The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers? (STEFANO MARKIDIS) explains why.
- “L-BFGS-B is currently the most critical technology for PINNs.”
- But! It has to be used in tandem with the Adam optimizer. (ie solution found by Adam and LBFGS refines the solution)
- Also, by nature, PINNs are usually underfitted as there is usually not enough data to learn from (for scale, a few TBs of data are considered ”enough”)

Some Comments

- Thank you for understanding and for fixing the excessive pixelation in the image originates. It may be attributed to various factors, such as graphical artifacts, color scale, irregular grid, and more. It is important to see the flow. – my apologies for the lack of proper plots. Please see the following slides for better ones.

- Are x and y and P normalized? - The directions are now normalized but the velocities are not

- The main flow is parallel to x, v and w are most of the time identically 0 except in the wake – indeed, we do notice that

- The anti-correlation of the vertical component is concerning – yes, hopefully with physics the predictions get better

- Interesting to follow the entire curve for both the training set and the testing set – will make a plot of this once the network is done learning

Some Comments

- The training phase is relatively time-consuming, and it appears to require few thousand epochs minimum. However, this extended training may lead to overfitting, as indicated by the significant degradation in performance on the validation dataset. - Agreed, with data loss only, that is definitely a problem but with physics loss that might not be such an issue. See following slides.
- The incorporation of physics (divergence and RANS) demonstrates a positive impact, enhancing almost all performance scores. This is an encouraging outcome in my opinion. – agreed!
- Vertical velocity (Velocity:2) emerges as particularly challenging to learn and generate. – yes, sometimes it does take longer to learn for a particular variable

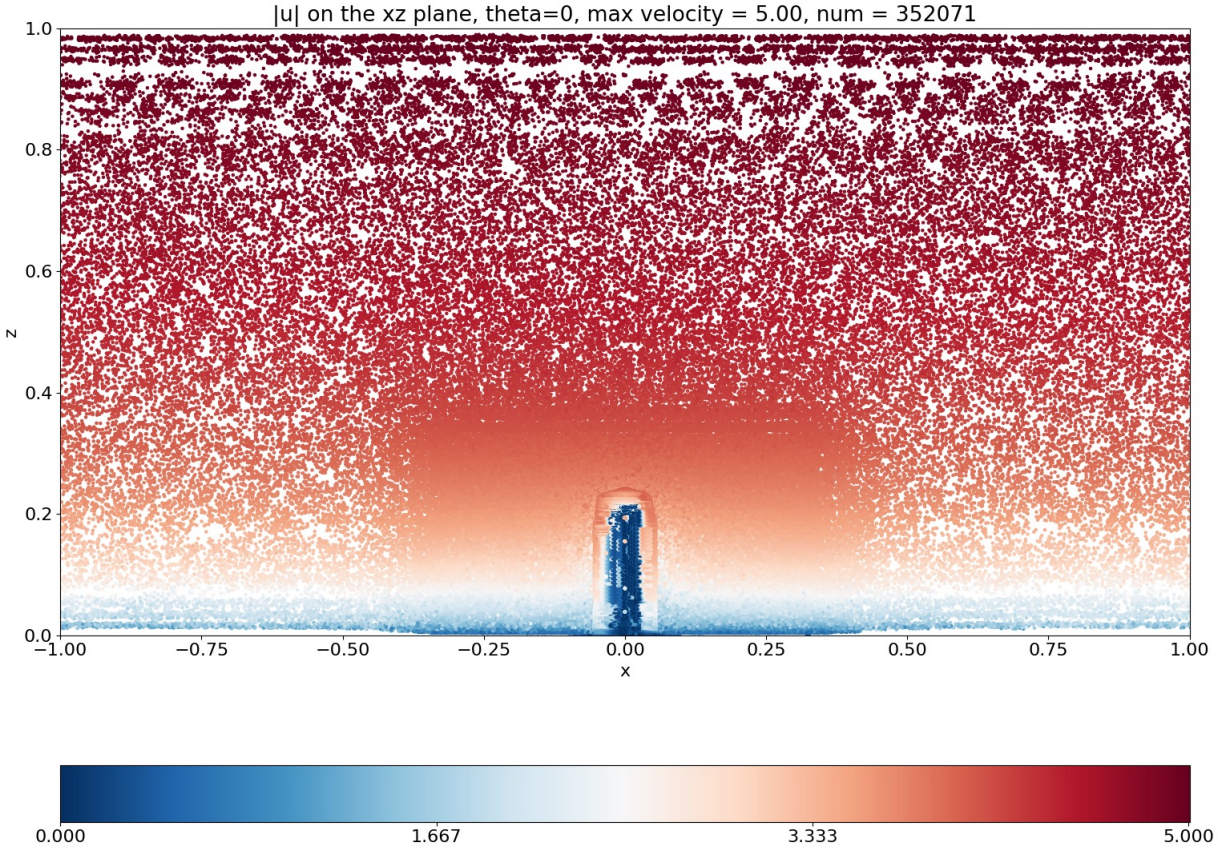
Some Comments

- The training phase is relatively time-consuming, and it appears to require few thousand epochs minimum. However, this extended training may lead to overfitting, as indicated by the significant degradation in performance on the validation dataset. - Agreed, with data loss only, that is definitely a problem but with physics loss that might not be such an issue. See following slides.
- The incorporation of physics (divergence and RANS) demonstrates a positive impact, enhancing almost all performance scores. This is an encouraging outcome in my opinion. – agreed!
- Vertical velocity (Velocity:2) emerges as particularly challenging to learn and generate. – yes, sometimes it does take longer to learn for a particular variable

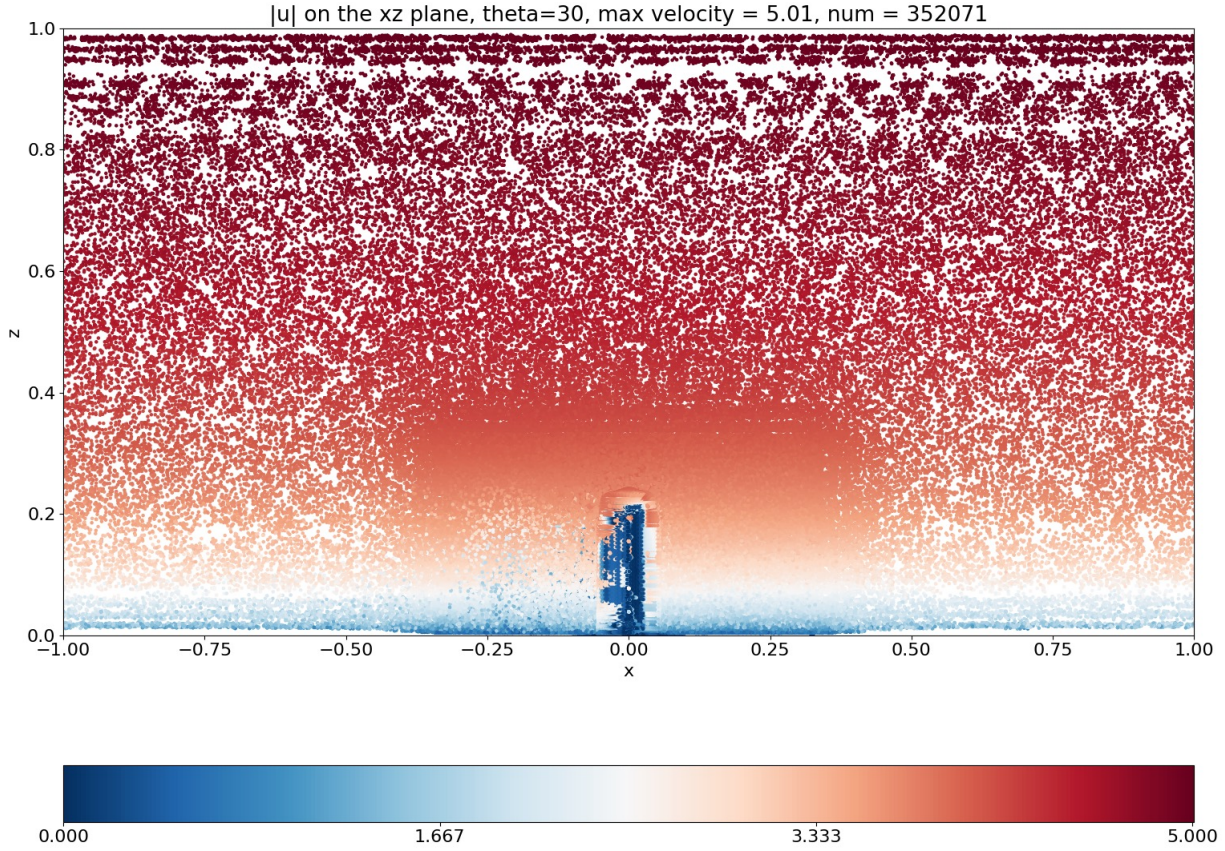
Some Comments

- Consider revisiting the graphical procedures of Wang Zhe and/or Bowen to create figures that offer an immediate visual representation of the flow patterns - done
- Verify the data sampling of both training and test datasets and ensure proper normalization of all variables to mitigate numerical artifacts - done
- To streamline the NN model, consider initially testing with outputs u, v, and w, and then progressively add variables such as nut, P, or others as needed? - agreed
- Explore ways to reduce the risk of overfitting, for instance, by experimenting with a lighter neural network architecture, such as reducing the number of neurons per layer from 128 to 64 for testing purposes.” – agreed
- Evaluate whether additional wind directions are necessary for a more comprehensive understanding of the flow dynamics – absolutely, if possible. DNS data would be welcome as well.

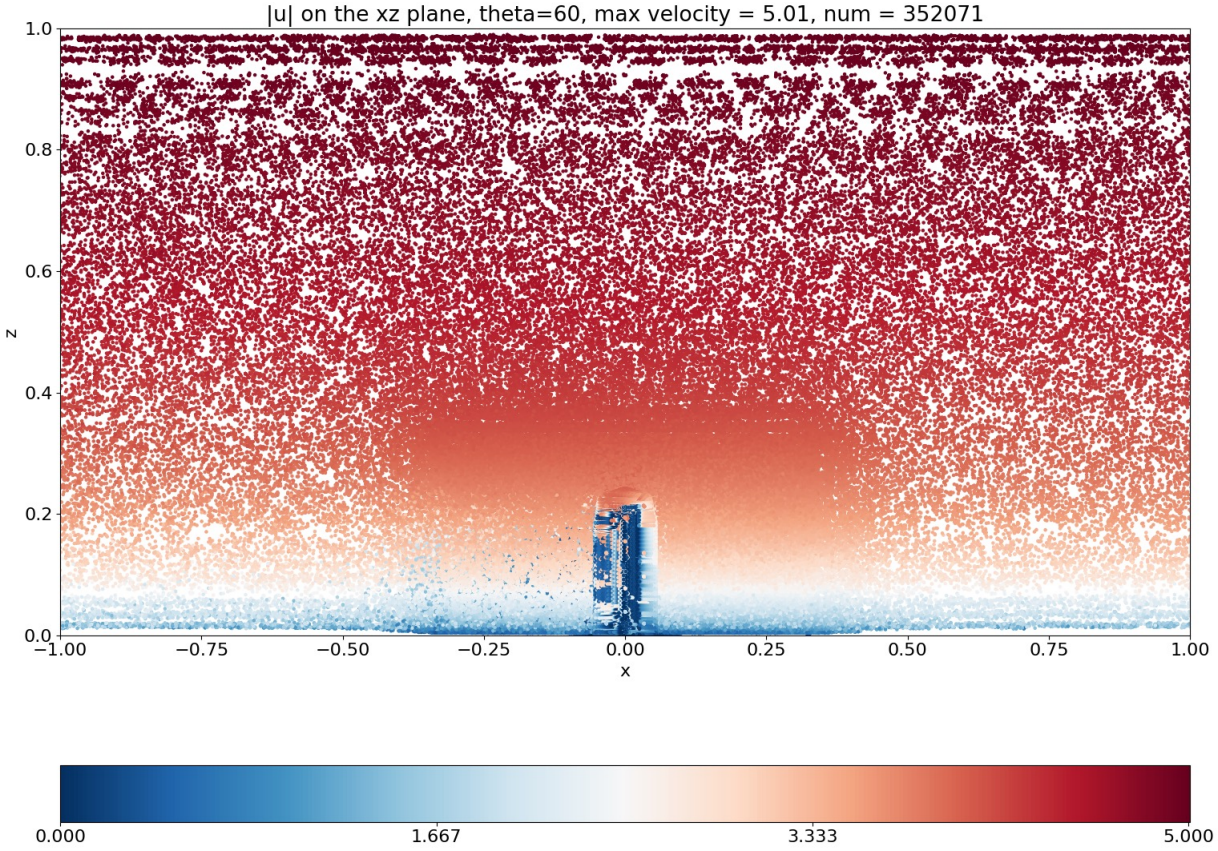
Data Plots – wind angle = 0



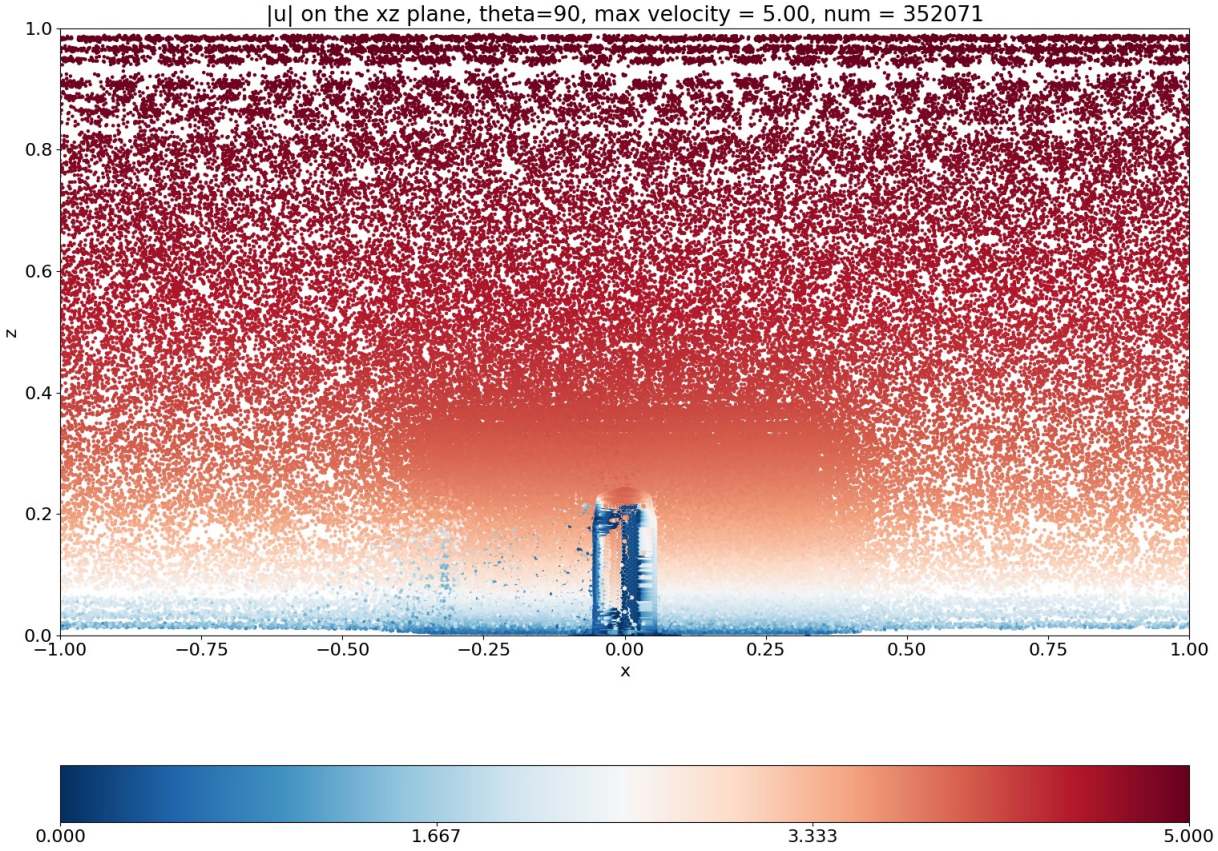
Data Plots – wind angle = 30



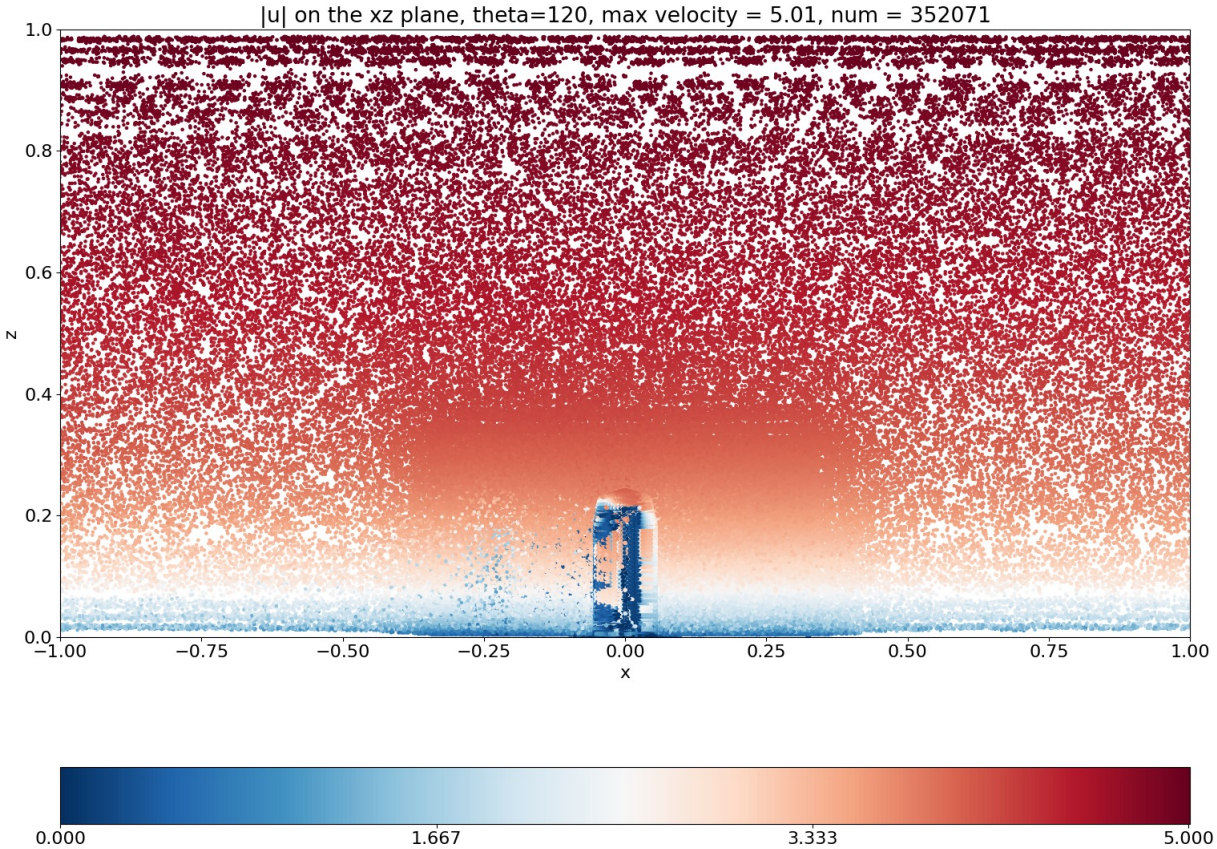
Data Plots – wind angle = 60



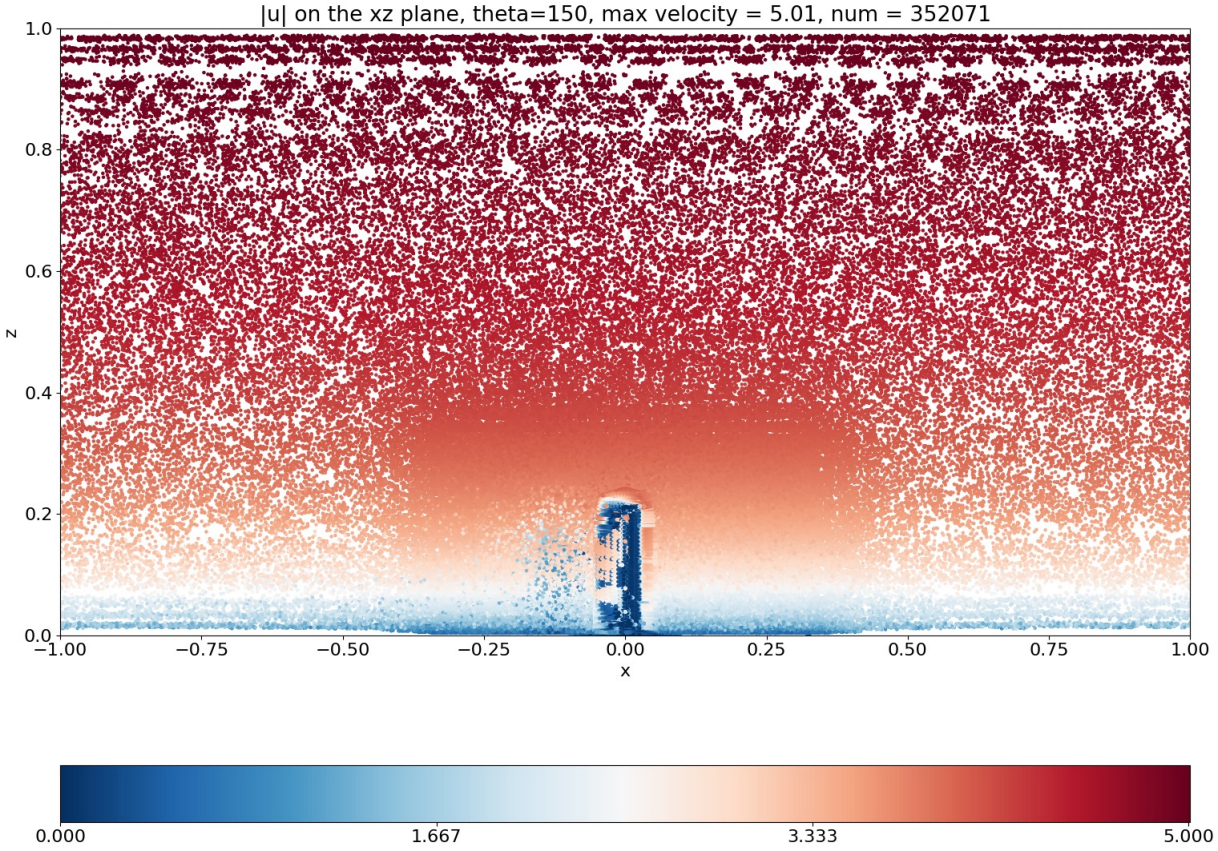
Data Plots – wind angle = 90



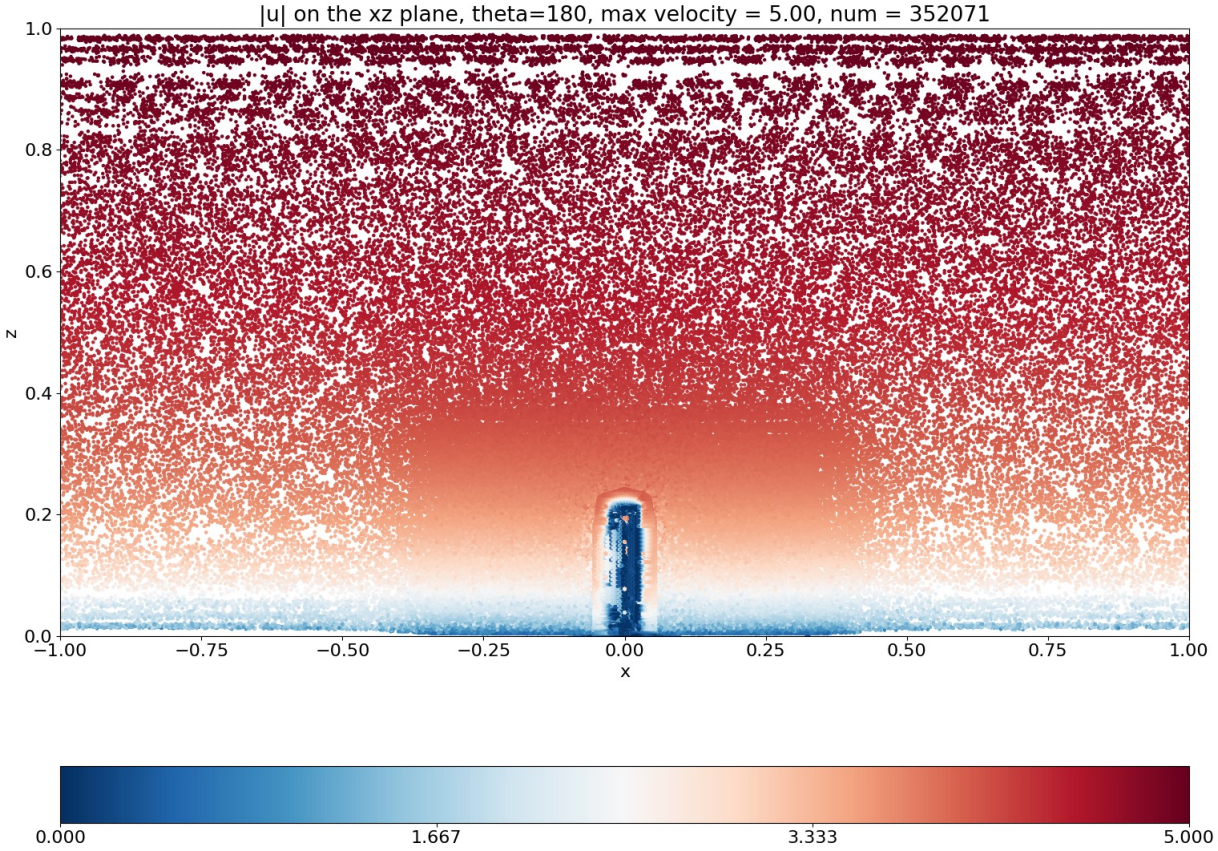
Data Plots – wind angle = 120



Data Plots – wind angle = 150



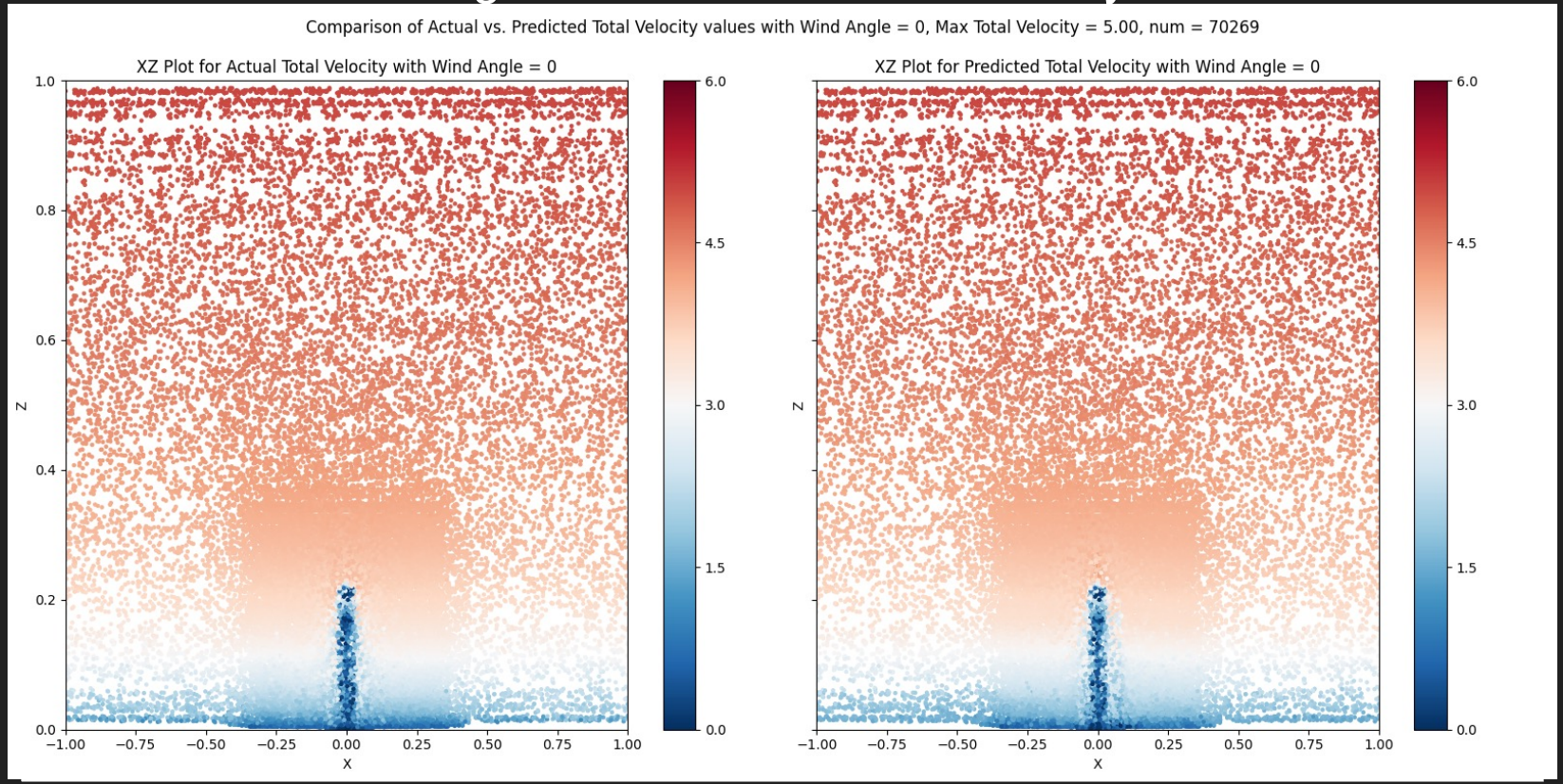
Data Plots – wind angle = 180



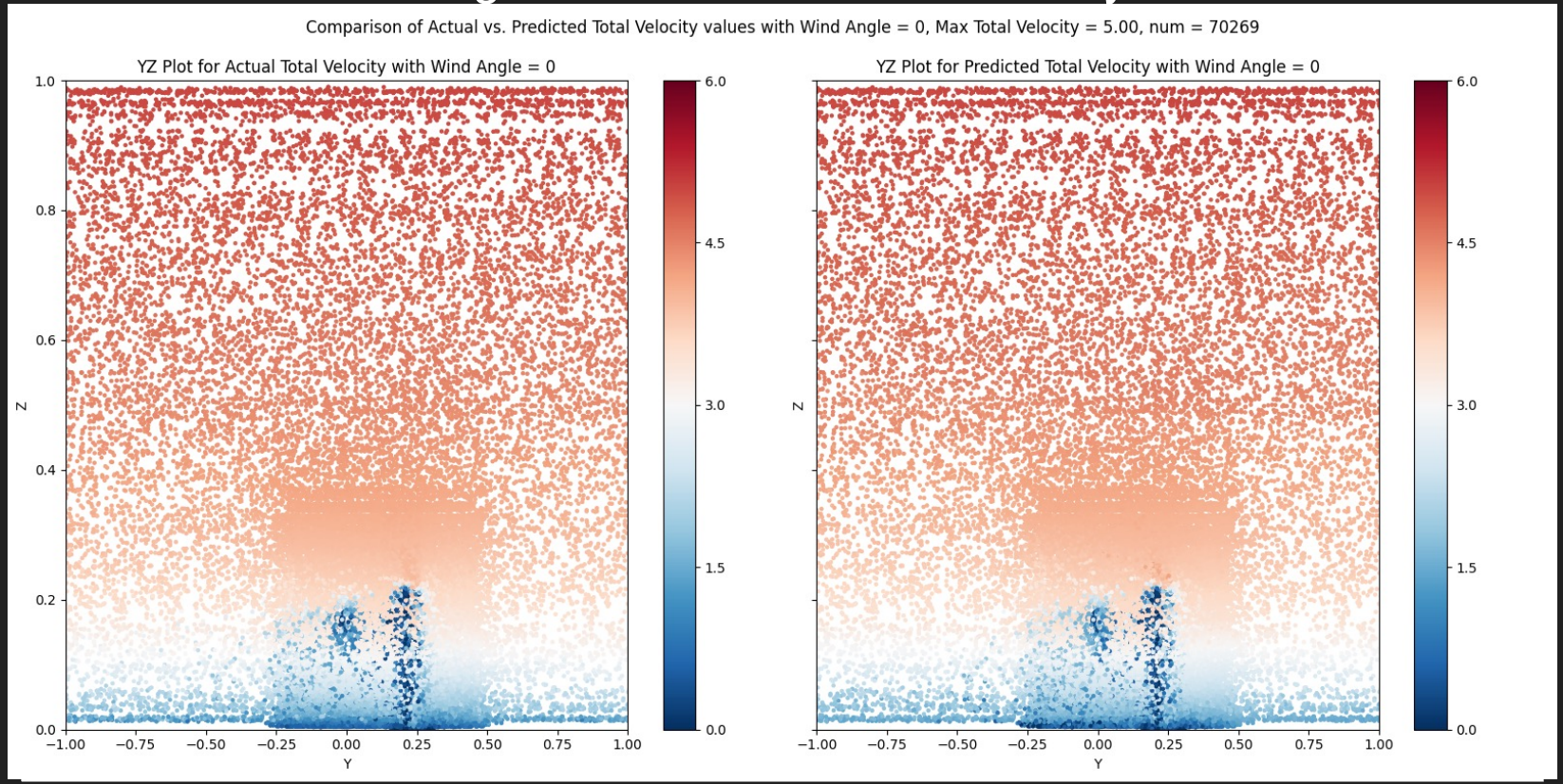
Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Testing Results - Metrics

Variable	MSE	RMSE	MAE	R2
Pressure	0.00801098	0.08950406	0.04117568	0.99199814
Velocity:0	0.008507	0.09223342	0.04432755	0.99150532
Velocity:1	0.0047879	0.06919463	0.03859898	0.99521592
Velocity:2	0.00714021	0.08449977	0.04616903	0.99285331

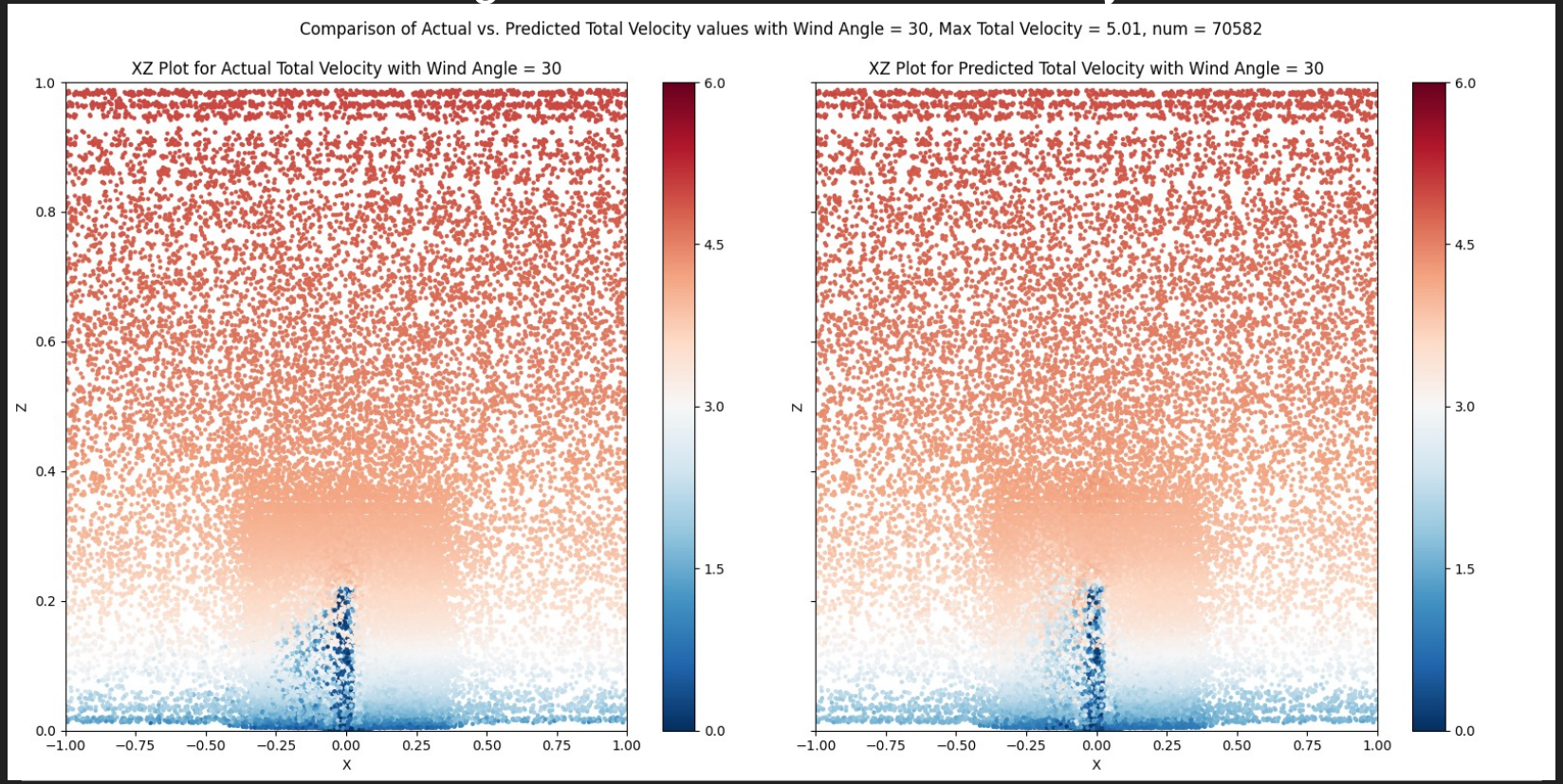
Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Testing Results - X-Z Total Velocity Plot



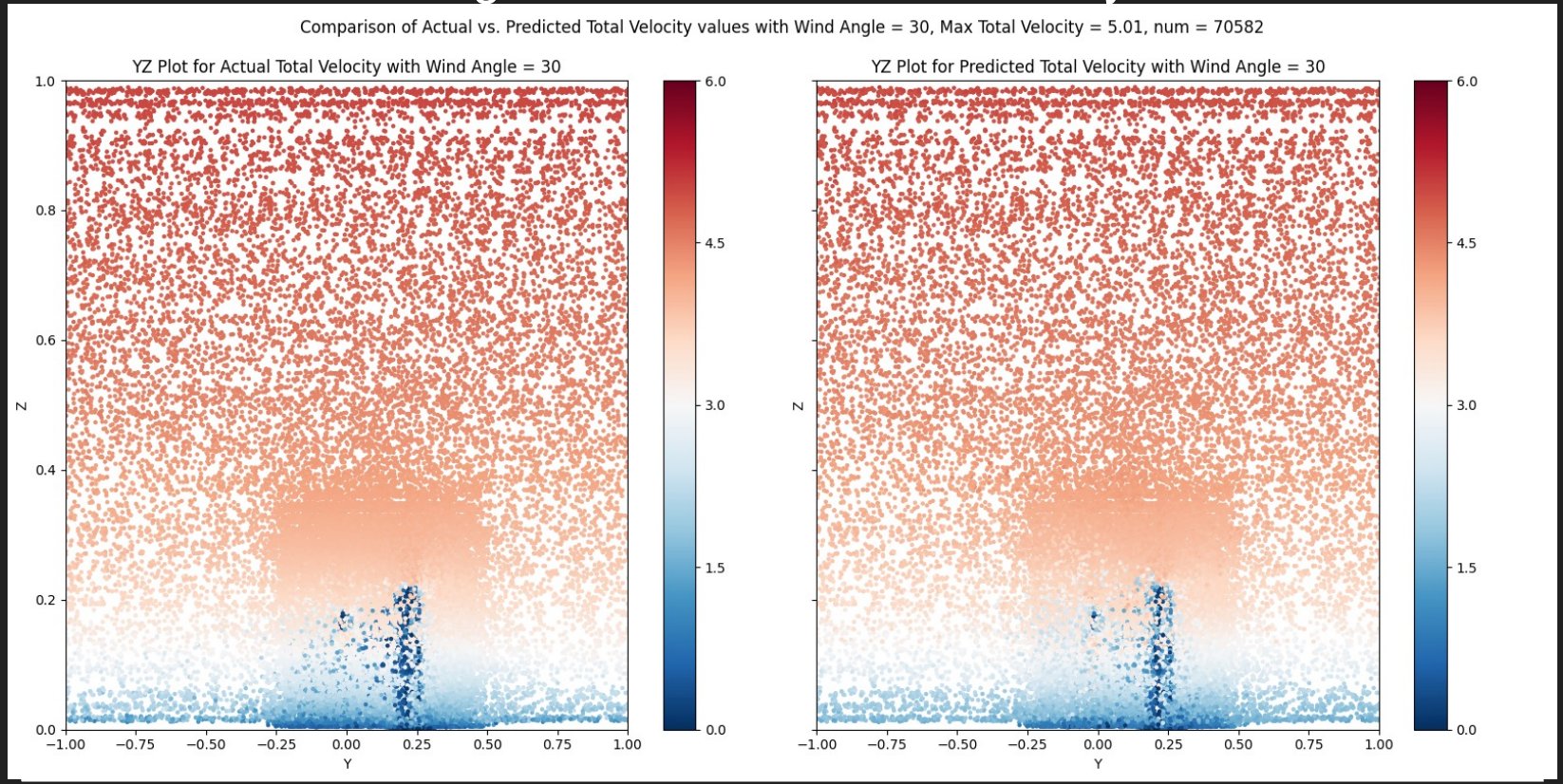
Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Testing Results - Y-Z Total Velocity Plot



Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Testing Results - X-Z Total Velocity Plot



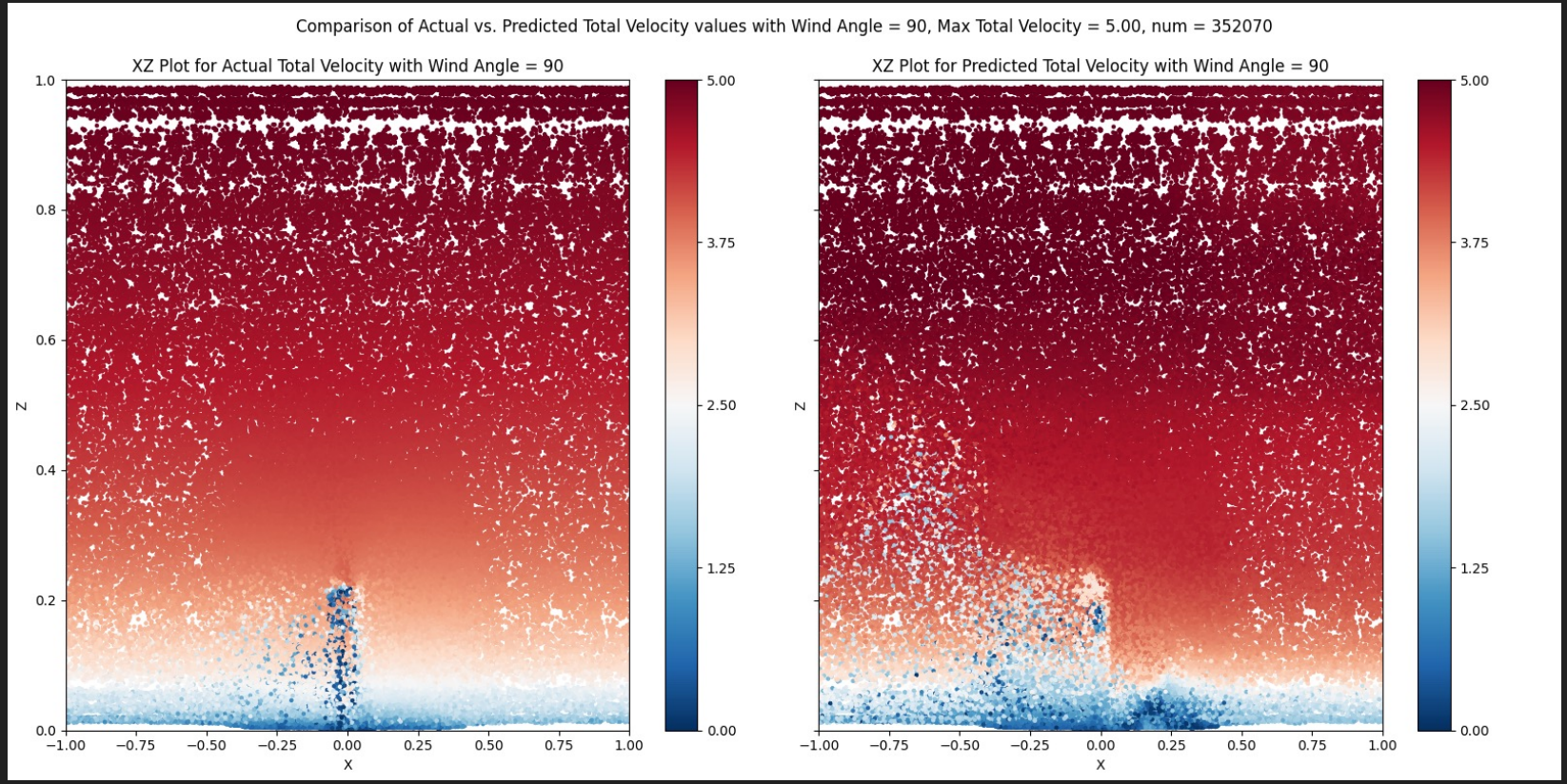
Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Testing Results - Y-Z Total Velocity Plot



Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Predicting Results - Metrics

Variable	MSE	RMSE	MAE	R2
Pressure	0.396712	0.6298508	0.34155917	-0.0734422
Velocity:0	0.46577922	0.6824802	0.3732554	0.52686685
Velocity:1	0.13879658	0.37255412	0.3009678	-21.283229
Velocity:2	1.167606	1.0805582	0.42440352	-0.0913069

Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Predicting Results - X-Z Total Velocity Plot



Progress so far - Data Loss + Cont Loss (Adam Optimizer) Threshold = 1E-5 (7680 Epochs, so far…), GPU Laptop Predicting Results - Y-Z Total Velocity Plot